2006-03

A survey and analysis of access control architectures for XML data

Estlund, Mark J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A SURVEY AND ANALYSIS OF ACCESS CONTROL ARCHITECTURES FOR XML DATA

by

Mark J. Estlund

March 2006

Thesis Advisor:	Cynthia E. Irvine
Second Reader:	Timothy E. Levin

Approved for public release; distribution unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: A Survey and Analysis of Access Control Architectures for XML Data			**5. FUNDING NUMBERS**
6. AUTHOR(S) Mark J. Estlund, Maj, USAF			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**
13. ABSTRACT (maximum 200 words) Extensible Markup Language (XML) has had a revolutionary effect on information technology. Both business and government have adopted XML as the format of choice for information sharing. Business uses XML to leverage the full potential of the Internet for e-Commerce. The government wants to leverage the ability to share information across many platforms between divergent agencies. In particular, in August 2004, Executive Order (EO) 13356 called for improved sharing of terrorist information to protect Americans.[1] XML provides a way to format information so that it is interoperable. The economic benefit of sharing data and resources is apparent. Sharing information between government agencies will assist in national security. However, there is still a requirement to control the flow and state of data. Therefore, access controls must be used to ensure data and information are protected. This thesis asks whether it is possible to provide a survey and analysis of how industry is enforcing access control on XML data, information, and documents that could serve as a foundation for XML security architectures for the government.			
14. SUBJECT TERMS XML, Access Control, XAMCL			**15. NUMBER OF PAGES** 66
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release, distribution unlimited

A SURVEY AND ANALYSIS OF ACCESS CONTROL ARCHITECTURES FOR XML DATA

Mark J. Estlund
Major, United States Air Force
B.A., University of Minnesota, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2006

Author: Mark J. Estlund

Approved by: Dr. Cynthia E. Irvine
Thesis Advisor

Timothy E. Levin
Second Reader/Co-Advisor

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Extensible Markup Language (XML) has had a revolutionary effect on information technology. Both business and government have adopted XML as the format of choice for information sharing. Business uses XML to leverage the full potential of the Internet for e-Commerce. The government wants to leverage the ability to share information across many platforms between divergent agencies. In particular, in August 2004, Executive Order (EO) 13356 called for improved sharing of terrorist information to protect Americans.[1] XML provides a way to format information so that it is interoperable. The economic benefit of sharing data and resources is apparent. Sharing information between government agencies will assist in national security. However, there is still a requirement to control the flow and state of data. Therefore, access controls must be used to ensure data and information are protected. This thesis asks whether it is possible to provide a survey and analysis of how industry is enforcing access control on XML data, information, and documents that could serve as a foundation for XML security architectures for the government.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First and foremost, I would like to thank my family for the love, support and sacrifices they have made during my time at the Naval Postgraduate School. Christy, Bryn, Haley, and Sam are the most important people in my life and I love you all. Next, I would like to thank my fellow NPS students who supported me along the way. LT Ken Gregoire, LT Ryan Ernst, LT Matt Mackay, Capt Curtis Smith, Maj David Stone, LT Les Sobol, LT Dan Speer, Ms Binh Duong, and Mr Mario Urrea. Finally, I would like to give special thanks to Dr. Cynthia Irvine and Prof Tim Levin. I was privileged to learn from and work with these outstanding educators and professionals. It is their guidance and leadership that allowed me to complete this journey.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Extensible Markup Language (XML) has had a revolutionary effect on information technology. XML is a standard for creating markup languages. With XML, it is possible to describe practically any type of information. What was once an idea to improve Standard Generalized Markup Language (SGML), has exploded into its own distinct field, weaving its way into every imaginable technology area including business, graphics, networking, and mobile technology, to name a few.

Both business and government have adopted XML as the format of choice for information sharing. Business uses XML to leverage the full potential of the Internet for e-Commerce. The government wants to leverage the ability to share information across many platforms between divergent agencies. In particular, in August 2004, Executive Order (EO) 13356 called for improved sharing of terrorist information to protect Americans.[1] XML provides a way to format information so that it is interoperable.

The language used in EO 13356 calls for the design and use of information systems, the timely dissemination of information, and creation of a means to allow access to other agencies' terrorist information. In response to the order, the Intelligence Community (IC) has established working groups in order to create standards by which information will be shared. Intelligence Information Sharing Standards (IISS) is a multi-phased program that uses XML based models to develop the standard by which all the IC agencies will create, edit, and share intelligence data.[2]

A. PROBLEM STATEMENT

The economic benefit of sharing data and resources is apparent. Sharing information between government agencies will assist in national security. However, there is still a requirement to control the flow and the state of data. Therefore, access controls must be used to ensure that data and information are protected.

XML's rapid development and expansion has left some areas unexplored, or underexplored. One area that has not received extensive review is access controls for XML-based data. Understanding how the commercial sector is addressing this issue is significant to the government and military since recent acquisition trends involving

information technology have been commercial-off-the-shelf-based (COTS). This thesis asks whether it is possible to provide a survey and analysis of how industry is enforcing access control on XML data, information, and documents that could serve as a foundation for XML security architectures for the government.

B. CHAPTER OVERVIEW

The thesis is composed of five chapters. The first chapter, the introduction, describes the motivation for the thesis. The second chapter, background, provides details necessary to understand XML, XML structure, how XML data is stored, how XML data is secured, and relevant XML standards. Chapter III examines four XML access control architectures. The architectures are reviewed for functionality, architectural design, how the products make access control decisions, and enforce those decisions, and to what scale and granularity the access control goes to. Chapter IV includes an extensive analysis and discussion of the four products. The fifth chapter presents the conclusions of the thesis.

II. BACKGROUND

Access based security is as old as civilization. A person's valuables, or *objects* were often physically separated for protection. It was the owner's discretion as to who gained access to those objects. Today, information and data are the valued objects that require protection. Databases can be likened to treasure chests, while firewalls and gateways are analogous to the walls around the castle and the drawbridge over the moat.

Extending beyond the visual analogy, there are many key components to the protection of information and data. A brief introduction to some of the key technology pieces will provide context for the description and analysis of current commercial solutions to XML access control, which comprise later chapters.

A. ACCESS CONTROL

A significant focus of this paper is access control. Therefore, a brief introduction to access control is warranted. Access rights are associated with objects. These rights permit operations that read of write the object, (e.g., read, write, or execute). Access controls are used to manage those access rights according to policy. Access controls add an additional layer of protection to an object beyond user identification and authentication to the system as a whole.

The two fundamental access control models are Discretionary Access Control (DAC) and Mandatory Access Control (MAC). Historically, many access control models have evolved from the Department of Defense efforts to prevent unauthorized access to classified information. The application of access control policies is now common in the commercial sector as well.[21]

DAC provides a run-time interface that allows modification of access rights to objects. The user who has control of the object is, for example, an owner. The owner has the "discretion" to then extend access to another user. If the owner has previously granted access to an object, that access can also be revoked.

MAC is defined in the DoD's Trusted Computer Security Evaluation Criteria (TCSEC) as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal

authorization (i.e., clearance) of subjects to access information of such sensitivity[20]." Use of a MAC system requires that all data objects containing information be given sensitivity labels (i.e., unclassified, secret, top secret) and usually compartment information. Therefore, a user must possess the proper clearances to gain access to an object. Additionally, the user does not have the ability to grant or revoke access to other users. This is because clearances of users and the classification of information are controlled by security administrators rather than the typical user.

Another means of administering access control is based on "roles". Role Based Access Control (RBAC) decisions are based on the positions, jobs, or responsibilities an individual user has as part of an organization. Users are grouped into a role and their access rights are based on that role. To express a policy using RBAC, it is important that the organization completes a thorough review of the defined roles and the rights associated with them. Once the roles and right have been defined, the roles serve as subjects would, regardless of whether it is a DAC or MAC system.

B. MULTILEVEL SECURITY (MLS)

An added dimension of security occurs when an information system contains resources at more than one security level. MAC can be applied in a system with different classifications resulting in *Multilevel Security (MLS).* Users with various security clearances are allowed to access the system concurrently, but the system only allows access to objects when a user possesses proper authorization. Therefore, if Alice has a secret clearance, then even though there may be "top secret" level information in the system, she would only be able to access secret and unclassified information. A benefit of an MLS system is that it alleviates the need for separate systems based on information classification. However, MLS systems are not risk-free. Physical security, inference risks, personnel security, and covert channel risks must be addressed. This paper will not deal with those issues, but they should be reviewed in the overall context of MLS systems.

C. INTRODUCTION TO XML, DATABASES, AND WEB SERVICES

1. Introduction to XML in Databases

In its simplest form, Extensible Markup Language (XML) is a way to describe or add special meaning to data, which can be as simple as text data or as complex as

network packets. Taking the definition one step further, XML is a markup language and a quickly evolving technology. While XML's original intent was to enable large-scale electronic publishing over the internet, its functionality is firmly rooted in its ability to describe and structure data. It is these qualities, along with XML's flexibility, platform independence and ease of use that has driven both the commercial sector's and government's vigorous adoption of XML for both data storage and web based data processing. Therefore, a general understanding of how XML is used with data storage and web services will be important to better understand XML's role in access control.

First, a quick tour of the main XML components will enable better understanding of the many XML related topics to follow. XML documents are made up, primarily, of *start tags, end tags, elements* and *attributes* An example of a start tag would be *<first_name>*. Every start tag must have a corresponding end tag. In the previous example, the matching end tag would be, *</first_name>*. Everything contained between, and including, the pair of tags creates an element, such as *<first_name>John</first_name>*. The text between the tags is referred to as the *element content.* Attributes are simple name/value pairs associated with an element, such as *<name nickname = "John-boy">*. An attribute is attached to the start tag, but not to the end tag. Lastly, XML documents have a hierarchical structure, as shown in Figure 1.

In this example the end tags have been left off for simplicity. The items in the document relate to each other in parent/child and sibling/sibling relationships. These descriptions will be the definition for XML components throughout this paper. While these definitions are simplified, they will be helpful when examining XML technology.

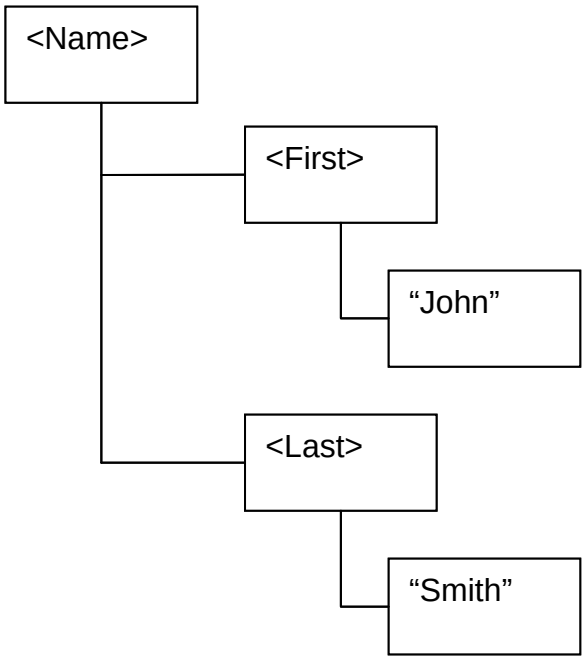

Figure 1. XML Tree Hierarchy

To understand how XML documents are stored in a database, it is necessary to understand that a document's structure plays an important role. There is a dichotomy in the structuring of XML documents, which determines what type of database is used and how that data is stored. Documents that are highly structured are said to be data-centric, while documents that are semi-structured or do not follow any structuring are considered to be document-centric.

Data-centric documents are characterized by somewhat predictable structure. Data tends to be more granular and rarely contains mixed content. The presentation of the data is consistent throughout the document. Data-centric documents are analogous to reference documents. They are documents that a human reader would scan for pieces of data. Examples of data-centric documents are a telephone book or as in the example shown in Figure 2, a sales order [3].

In this example of a data-centric document, every sales order follows a well-structured "recipe". Each sales order has exactly one customer, and the customer has specific data associated with it. Each item on the sales order has a part number, description, and price.

The data structure translates well to storage in a database or conversely, data from a database could be used to form a structured data-centric document.

```
<SalesOrder SONumber="12345">
      <Customer CustNumber="543">
         <CustName>ABC Industries</CustName>
         <Street>123 Main St.</Street>
         <City>Chicago</City>
         <State>IL</State>
         <PostCode>60609</PostCode>
      </Customer>
      <OrderDate>981215</OrderDate>
      <Item ItemNumber="1">
         <Part PartNumber="123">
            <Description>
               <p><b>Turkey wrench:</b><br />
               Stainless steel, one-piece construction,
               lifetime guarantee.</p>
            </Description>
            <Price>9.95</Price>
         </Part>
         <Quantity>10</Quantity>
      </Item>
      <Item ItemNumber="2">
         <Part PartNumber="456">
            <Description>
               <p><b>Stuffing separator:<b><br />
               Aluminum, one-year guarantee.</p>
            </Description>
            <Price>13.27</Price>
         </Part>
         <Quantity>5</Quantity>
      </Item>
   </SalesOrder>
```

Figure 2. Data-centric example

(From http://www.rpbourret.com/xml/XMLAndDatabases.htm)

Documents that do not follow such strict templates are described as semi-structured or document-centric. These documents are not consistent from one to another in size and content. Document-centric products are also described as being less granular and fragmented. Additionally, these documents often contain large amounts of mixed content. Examples of semi-structured documents are advertisements, procedures in a manual, and glossary-entries. Document-centric documents are meant to be human readable. The example in Figure 3 demonstrates a product description [3].

```
<Product>

   <Intro>
   The <ProductName>Turkey Wrench</ProductName> from <Developer>Full
   Fabrication Labs, Inc.</Developer> is <Summary>like a monkey wrench,
   but not as big.</Summary>
   </Intro>

   <Description>

   <Para>The turkey wrench, which comes in <i>both right- and left-
   handed versions (skyhook optional)</i>, is made of the <b>finest
   stainless steel</b>. The Readi-grip rubberized handle quickly adapts
   to your hands, even in the greasiest situations. Adjustment is
   possible through a variety of custom dials.</Para>

   <Para>You can:</Para>

   <List>
   <Item><Link URL="Order.html">Order your own turkey
wrench</Link></Item>
   <Item><Link URL="Wrenches.htm">Read more about
wrenches</Link></Item>
   <Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
   </List>

   <Para>The turkey wrench costs <b>just $19.99</b> and, if you
   order now, comes with a <b>hand-crafted shrimp hammer</b> as a
   bonus gift.</Para>

   </Description>

   </Product>
```

Figure 3. Document-centric example

(From http://www.rpbourret.com/xml/XMLAndDatabases.htm)

In this example, the document still utilizes tags to organize data, but the content and the amount of content contained between the tags is variable. Most semi-structured documents are written by hand in XML or some other format that can be converted to XML. While document-centric products are human readable, this makes them difficult to interface with traditional databases.

D. XML DATABASE CATAGORIES

The difference between structured and semi-structured documents is enough to warrant separate and distinct data storage strategies. This results in two main models of XML databases. The first model, called *XML-Enabled*, is a relational database and is used with data-centric documents. Document-centric products, however, require XML-

specific databases, know as *Native XML* databases. Additionally, object-oriented databases have the ability to store XML documents. However, it is more common to store XML data as objects in a relational database. Therefore, object-oriented models will be given limited attention.

The following sections will briefly introduce the most common database models and address how they work with XML data. The descriptions will conclude with a short comparison of the models. The discussion will then move to several other database products that are designed specifically for XML. This will provide necessary background before discussion and analysis of access control strategies and security concerns.

1. XML-Enabled Databases

An XML-Enabled database is essentially a common relational database, but with additional functionality to work with XML data. Relational databases have traditionally been used to store structured data. With the increasing popularity and usefulness of XML, relational database developers have built XML processing capabilities into their products. XML-Enabled databases are best used with highly structured and granular data. Working with structured data allows for clearer translation between XML schema and database schema. When using relational databases to store XML data, it is difficult to cleanly create a table schema on the fly. Therefore, XML-Enabled databases are said to be *schema-dependent*. If XML documents are consistent, well-structured, and predictable, a proper database schema can be developed. Most XML-Enabled products require the use of *document type definitions* (DTD) or style sheets to ensure correct mapping between the XML document and the database. Additionally, XML middleware products may be introduced to assist with mapping and translation. Middleware products may also be used with Native XML databases. Further detail on middleware will be provided in section C.2.

While relational database technology is widely accepted and most often the default choice of users, the mapping between relational databases and XML documents are not quite a “hand-in-glove” fit.[4][5] The essential organization of XML documents is hierarchical, while relational databases flatten things out. Therefore, the XML document must be pulled apart to store the data. There are actually several approaches to accomplish the mapping. As addressed previously, the use of DTD’s to define a schema

is a common method. A less exact method is for a database administrator to manually create the database schema based on expected elements and attributes. Another potential design method is to develop a mapping based on expected query workload.[6] For the purposes of this paper, we will focus on mappings performed with DTD's or stylesheets.

In most cases when decomposing an XML document to be stored in a relational database, the result is a set of relational tables called an *XML collection*. IBM refers to this as *shredding*.[7] The key considerations to storing XML data in this manner are, (1) only the element content, i.e., the data between the tags, is stored and (2) the tables schema is based on the document's elements and attributes. Since the data is now stored in a relational manner, regular SQL statements can be used to query data, create views and make updates to data. Data retrieval presents possible challenges. To retrieve, or compose a complete XML document from the stored data, a complex set of joins must be developed. Additionally, there is a risk of data loss when attempting to recreate a document. This is due to the fact that the document, when shredded, was most likely separated into multiple tables. If it is desired to create an original document from stored data, there is still a necessity to work with joins.

2. Native XML Databases

Native XML databases specialize in storing XML documents. The database focuses on the structure of the document as opposed to the data in the document. The database defines a storage model based on the elements and attributes of the document. Therefore, the document is the fundamental unit of storage, whereas in a relational database the rows of a table are it's fundamental unit of storage. Since the entire XML document is stored as one unit, the structure of the document remains intact. Succinctly put by Kimbro Staken, co-founder of the XML:DB Initiative, "Documents go in and documents come out." [5] Native XML databases are also well suited to store collections or sets of documents. Previously, we compared an XML document with a row or tuple of a relational model. Continuing with this analogy, a collection of XML documents make up a set that can be manipulated or queried, much like a collection of tuples make up a relational database table. A distinct difference between relational and Native XML databases is that while a relational database requires a predefined schema to define the fields of the tuples, no such schema is required in the latter model. It should be noted that

validation against a schema or DTD can occur in many Native XML databases, but this function is not essential to its operation. This *schema-independent* quality introduces a larger degree of flexibility. There is, however a drawback, low data integrity, in the sense that there is no guarantee of a well formed documents or that the data in the document is usable. Recall that a schema or DTD regulates not only the structure of the document, but the type of data inserted in elements and attributes. If schema structure is a major concern, then it is necessary to ensure the product to be used can support this.

We have briefly addressed the storage aspects of XML data in a Native XML database. How does "store-as-a-document" methodology affect querying and retrieval of information? First and foremost, Native XML databases do not work with Standard Query Language (SQL). SQL is not designed to query hierarchical structures such as those in an XML document. To complete queries against Native XML databases, XML-specific query languages must be used. The current W3C standard XML query language is called XPath [8]. A more robust standard, XQuery, is currently in beta testing and should become a W3C recommendation in the near future [9]. A more detailed description of XML query languages and query engines will follow in section C.4.

There are three perspectives to data retrieval possible when working with a Native XML database: document retrieval, extracting specific facts, and word searches. As explained before, a Native XML database stores the entire document as a unit. Therefore, retrieving an exact duplicate of that document is trivial. Queries can be based on unique references or combinations of properties that documents must possess. Specific information can also be retrieved from a stored document, or a set of stored documents. Queries for specific facts operate on the logical structure of an XML document. A document is viewed as a tree of nodes, and therefore queries follow those branches to the requested information. The final method, searching for "key" words, is ultimately a text search. While the desire is that data is effectively marked-up, much of the document centric data is "chunks" of text. Linear searching through text is not an efficient method of data retrieval, and various indexing techniques may be applied to optimize the searching.

3. Object-Oriented Databases

Riding the coat-tails of object-oriented programming, object-oriented databases (or just object databases (ODBMS)) gained most of their popularity in the 1980's. ODBMS's followed a hierarchical structure. When working with XML documents, mappings were based of *classes* of data. Each class could contain objects that were used to transfer data from the XML documents to the database. However, the continued popularity and ease of use of relational databases relegated ODBMS's to niche markets.[13] There are no indications of performance problems storing XML data on a ODBM's, however the lack of products on the market resulted in ODBMS's exclusion in this survey of XML databases.

E. ADDITIONAL XML DATBASE TECHNOLOGIES

While the focus of this chapter is about the actual database products and how they work with XML, there are additional products that support the relationship between databases and documents. A full survey of these products would be quite substantial. Therefore, this section will provide a brief introduction to these products.

1. XML Servers

XML servers are most commonly web application servers, or custom servers. Uses vary from building distributed applications to publishing XML documents on the web.

2. XML Middleware

Middleware is software that provides both an interface to the database, as well as, an interface for tools that create XML documents or add XML data to the database. Middleware is used to transfer data between an XML document and a database, or vice versa. This software allows a user to build a XML document directly from SQL query results, or to extract information from an XML document in order to update a database (as shown in Figure 4). Middleware is most often used with relational databases. Figure 4 shows a notional view of an XML system that uses middleware.

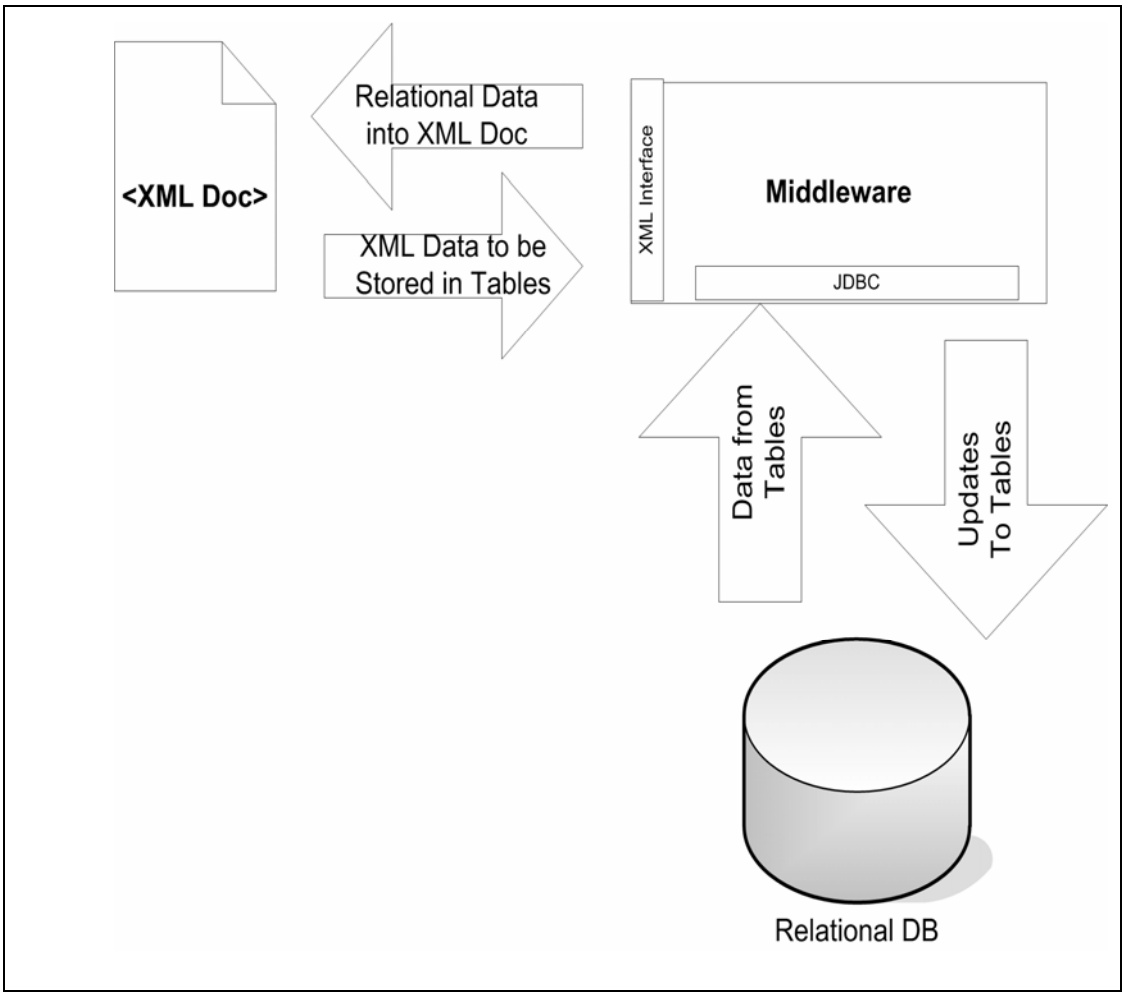

Figure 4. XML Middleware example

3. XML Wrappers

Wrappers are software products that treat XML data like relational data. The term, which comes from federated database systems, means that a translation module forms a new interface to a system so its data (e.g., XML) is presented in the desired model (e.g., relational). The *wrapped* XML data can then be transferred to or from a particular data source using SQL statements. SQL queries (e.g., SELECT statements) can also to be performed to search through an XML document.

4. XML Query Engines

XML Query Engines are stand-alone programs used to query XML documents and data. These engines are typically used when working with Native XML databases. Historically, the functionality of XML queries was limited to a single document. Products are now being developed with the capabilities to extract data from a single

documents or collections of documents. Furthermore, products such as XQuery have the ability to work with data locally, or across the Internet. This allows for interaction between the web and XML databases.[9]

5. Content (Document) Management Systems

Content Management Systems are applications used in conjunction with XML databases. As the name suggests, the purpose of the application is to provide an interface to a database and manage its content. These systems break XML documents into fragments and then store them in a database. Users then retrieve fragments from the database to produce new documents. Publishing and version control functionalities are their main selling points, while features like multi-user access are also desirable. Generally, these applications are transparent to the user.[3]

6. XML Data Binding Products

XML data binding is the binding of XML documents to objects designed for the data in those documents. This binding allows applications that are usually data-centric to work with the data that has been "serialized" as XML. Additionally, the binding allows the XML schema to map to an object schema and vice versa. This mapping allows XML documents to be broken into objects for storage in a database, or allows the objects to be retrieved from the database and used to create an XML document. A limitation of data binding is the potential for loss of information. XML attributes, elements, text, and the relationship between them are maintained, however, comments, entity references, and additional information are not.[3]

F. XML SECURITY

As XML has developed and its use has grown, it has been recognized that security features are needed. While existing Internet technologies, such as Secure Sockets Layer and username/password authentication provide a level of security for the transmission of data, additional functionality is required once the data is received at the server. Securing the data itself, as opposed to only its transport, adds an additional requirement for security. Digital signatures and encryption of metadata are used for transmitting and storing XML in a secure manner.[19]

1. XML Digital Signatures (XML-DSig)

Digital signatures, in general, have the capability to provide data integrity, authentication, and non-repudiation when used properly. This is accomplished through the use of a public and private key pair, and a hash of the plain text. A user creates a hash of the plain text and then encrypts the hash with his private key. This creates the *digital signature*. The user then sends the signed hash and plain text to its destination (the signed hash and plain text may be encrypted again, using the receiver's public key for confidentiality). Once received, the plain text is hashed, the signed hash is decrypted using the sender's public key (to verify the message actually came from the sender), and the hashes are compared. If the hashes match, integrity has been verified.

XML Digital Signatures (XML-DSig) provide those features for XML documents, or portions of XML documents. The ability of XML-DSig to sign specific portions of the XML tree, versus then entire document, is a fundamental feature.[19] This function can guarantee the integrity of one portion of a document, while leaving other portions open for changes. Those additions or changes can then be signed as well by another user. An example would be when a document has many authors contributing at different times. Each person completes his portion, digitally signs the portion, and then forwards the entire document to the next author.

To perform a digital signature on an XML document, the user first must identify what content (i.e., the data object) is to be signed. Then a hash of the data object is computed, and the resulting value is placed in an element. Recall that an element is composed of start and end tags with element content in between (element content can be other elements or attributes, as well). Next, the contents of that element are digested and cryptographically signed. The digital signature is represented in the *Signature* element, as shown in Figure 5. The *Signature* element is referenced back to the data object via a URI. There is only one *Signature* element for any data object signed.

```
<Signature ID?>
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod/>
       (<Reference URI? >
         (<Transforms>)?
         <DigestMethod>
         <DigestValue>
       </Reference>)+
     </SignedInfo>
     <SignatureValue>
    (<KeyInfo>)?
    (<Object ID?>)*
   </Signature>
```

Figure 5. XML Digital Signature

(From http://www.w3.org/TR/xmldsig-core/#sec-Overview)

This is denoted by the "?" in the element. The "?", "+", and "*" are cardinality indicators, where the "?" means the element may appear zero or one time; the "+" means the element may appear one or more times; and the "*" means the element may appear zero or more times. If included as part of the XML document, signatures are related to local data objects via fragment identifiers.

2. XML Encryption (XML Sec)

The counterpart to XML-DSig is XML Encryption (XML-Sec). XML-Sec, like traditional cryptography, is used to conceal information. Encrypting an entire XML document is actually quite straightforward. It is when a portion of an XML document is required to be encrypted, that the added value of XML and XML-Sec are realized. If an XML document is authored by different people with different authorizations for various parts of the content, there may be cause to encrypt portions of data. A doctor or researcher for example, may need to view a patient's medical history, but has no need to see a patient's insurance information. Conversely, a hospital administrator would need access to the patient's insurance information, but not the patient's medical history.

In addition to selectively signing specific elements of an XML document, XML-Sec supports the ability to encrypt the element tags themselves. With this added feature comes potential for added problems. Encrypting tags can undermine XML's strength for searching through documents using DTD's or schemas. Additionally, possessing DTD's

or schemas for a document with encrypted tags creates a risk of a plain text attack on the cryptography. If an attacker possesses the plain text (via the DTD or schema) and the encrypted document, he may be able to break the cipher. This can compromise the confidentiality of future documents. The working draft of XML-Sec at W3C is addressing these and other potential security shortfalls.[25]

When an element and its contents are encrypted, they are replaced by an <EncryptedData> element and reference to the cipher data. The XML code in Figure 6 shows an unencrypted XML document with a person's credit card information. The code fragment in Figure 7 shows how the document is changed after encrypting the details of the credit card.

```
<?xml version='1.0'?>
       <PaymentInfo xmlns='http://example.org/paymentv2'>
         <Name>John Smith<Name/>
         <CreditCard Limit='5,000' Currency='USD'>
           <Number>4019 2445 0277 5567</Number>
           <Issuer>Bank of the Internet</Issuer>
           <Expiration>04/02</Expiration>
         </CreditCard>
       </PaymentInfo>
```

Figure 6. Unencrypted Credit Card Information for John Smith

(From http://www-128.ibm.com/developerworks/security/library/s-xmlsec.html)

```
       <PaymentInfo xmlns='http://example.org/paymentv2'>
         <Name>John Smith<Name/>
         <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
          xmlns='http://www.w3.org/2001/04/xmlenc#'>
             <CipherData><CipherValue>A23B45C56</CipherValue></CipherData>
         </EncryptedData>
       </PaymentInfo>
```

Figure 7. Encrypted Credit Card Information

(From http://www-128.ibm.com/developerworks/security/library/s-xmlsec.html)

In this example, all the information within the <CreditCard> element has been encrypted and replaced by encrypted elements and cipher elements. This simple example provides a good example of XML-Sig's ability to protect information.

G. WEB SERVICES, SAML, AND XACML

With the advent of the Internet, businesses were able to more efficiently exchange information and data that was previously isolated. However, this required connected systems to be interoperable and connections via the Internet included some vulnerabilities and security concerns. There was a need for efficient information and data exchange, coupled with the desire for a more secure, *Intranet* feeling. Web Services is one such solution. Its advantages are that it is based on HTTP protocol and it uses XML as its base language.[22] These two factors aid in development ease, interoperability, and portability.

Web Services uses a variety of protocols to complete information and data exchanges. Simple Object Access Protocol (SOAP), Universal Discovery, Description, and Integration protocol (UDDI), and the Web Services Description Language (WSDL) are necessary pieces of the puzzle to complete a transaction. SOAP is the protocol that allows objects on one computer to call and make use of objects on other computers, and otherwise exchange information over the Internet using HTTP. SOAP messages are formatted in XML. UDDI is a protocol which allows Web Services to be registered so they can be looked up or discovered by users or other Web Services. WSDL is an XML-based language through which different services are described in the UDDI. Additionally, WSDL provides guidance on the structure and format of requests made. While these protocols made the information and data exchanges possible, there were still security issues to be addressed.

1. Secure Assertion Markup Language (SAML)

SAML is an XML-based security specification for exchanging authentication and authorization information. An *assertion* is a declaration of facts or statements about a subject (typically authentication and authorization information), such as has been described for “capability” systems.[28] The assertions are the basis for access to Web Services. Information that is common to all assertions is:

- Issuer and issuance timestamp
- Assertion ID

- Subject: Name, security domain, and possibly a public key for confirmation
- Conditions that satisfy a valid assertion (e.g., time transactions are allowed, role based restriction, domain restriction)

```
      <saml:Assertion
       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
       MajorVersion="1" MinorVersion="1"
       AssertionID="..."
       Issuer="https://idp.org/saml/"
       IssueInstant="2002-06-19T17:05:37.795Z">
       <saml:Conditions
         NotBefore="2002-06-19T17:00:37.795Z"
         NotOnOrAfter="2002-06-19T17:10:37.795Z"/>
       <saml:AuthenticationStatement
         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
         AuthenticationInstant="2002-06-19T17:05:17.706Z">
         <saml:Subject>
           <saml:NameIdentifier
             Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
             user@mail.idp.org
           </saml:NameIdentifier>
           <saml:SubjectConfirmation>
             <saml:ConfirmationMethod>
               urn:oasis:names:tc:SAML:1.0:cm:artifact
             </saml:ConfirmationMethod>
           </saml:SubjectConfirmation>
         </saml:Subject>
       </saml:AuthenticationStatement>
      </saml:Assertion>
```

Figure 8. SAML Assertion

(From http://en.wikipedia.org/wiki/SAML)

Each of the required components can be identified in Figure 8. For instance, the *issuer* is https://idp.org/saml/. This example uses time restrictions, *NotBefore* and *NotOnOrAfter*. SAML has limited capabilities to provide access control to data objects. This has created a requirement for more fine-grained access control. XACML, Extensible Access Control Markup Language, is an access control policy language that is quickly being called to fill that role.

2. XACML

Extensible Access Control Markup Language (XACML) is a new markup language defined by an OASIS Technical Committee (Version 2.0 was approved

February 2005). XACML "can be viewed as a basic specification of a policy server that provides fine-grained access control within a Web Services environment."[24] The XACML standard defines the access control syntax and semantics, as well as, provides an architectural framework in which it is processed. An added benefit of XACML is its ability to interoperate with other systems, where typically each application had its own access control scheme.

XACML access control policies are written in XML and stored for later reference. The rules define permitted or non-permitted actions for subjects on a resource (or object). In Figure 9, the access control rule says "Permit John to open the door."

```
<Rule
    RuleId=""
    Effect="Permit">
  <Description>John can open the door.</Description>
  <Target>
      <Subjects>
          <Subject>
              <SubjectMatch
                    MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                  <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
                  <SubjectAttributeDesignator
                     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                        DataType="http://www.w3.org/2001/XMLSchema#string"/>
              </SubjectMatch>
          </Subject>
      </Subjects>
      <Resources>
          <Resource>
              <ResourceMatch
                    MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
                  <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#anyURI">door</AttributeValue>
```

```
                        <ResourceAttributeDesignator
                         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                            DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
                    <ActionMatch
                          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">open</AttributeValue>
                        <ActionAttributeDesignator
                             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                              DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
```

Figure 9. XACML Access Control Rule

(From www.idealliance.org/papers/dx_xmle04/papers/04-01-04.html)

In general, a rule can have the effect of permitting or denying access to a resource. In this example, the Effect attribute in the Rule element defines the effect as "Permit". The Subject, Resource and Action elements constrain the rule to a specific subject, resource, and action (in this case permitting John to *open* the door.) Finer grained policies can be incorporated into the Actions elements (such as read or write permissions.)

XACML additionally defines the processing environment that utilizes policies and enforces the access control decisions. There are two main components to this environment, the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP).

Figure 10 depicts a conceptual view of the environment and how an access control decision is made. The PEP receives a request from a user and generates its own request based on subject, resource, and action attributes. This request goes to the PDP where it is processed against policy. The PDP then returns an access control response to the PEP. If a match has been made, the user's original request is allowed. If a match has not been made, the request is denied.

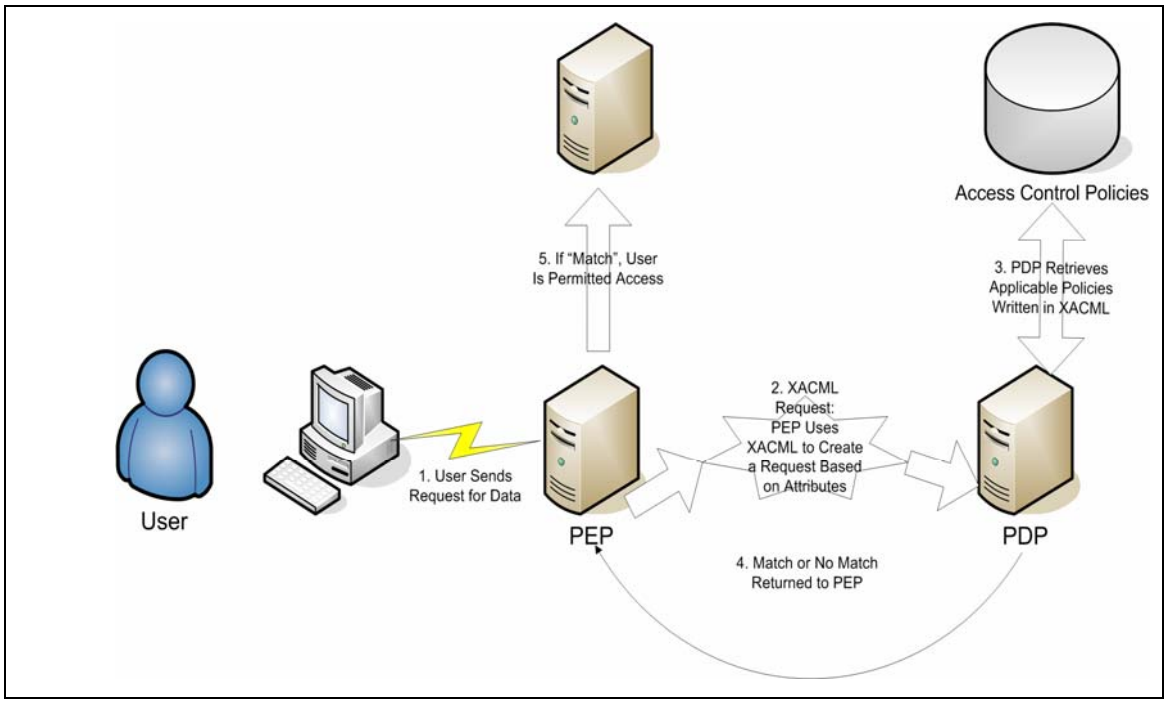

Figure 10. XACML Processing Environment

SAML and XACML are key components to secure data and information exchange using Web Services. This has been a high level description of Web Services, SAML, and XACML. Significant research and investigation could be conducted on any of these subjects.

As there are many ways in which XML information and data are stored and transmitted, there are also a variety of ways in which they are protected. While XML access control is still in its infancy, there are several products that provide, or claim to provide, access control. In the following chapter, four commercial XML access control products will be reviewed.

III. XML ACCESS CONTROL ARCHITECTURES

Government's reliance on commercial off the shelf (COTS) products and vender-partnering establishes a need to know what industry has accomplished in the area of XML access control. The commercial sector's desire to leverage the benefits of XML (platform independence and interoperability across applications) is a driving force behind the development of new XML technologies and standards. One area of XML technology rising in importance is XML access control. However, as is common in industry, every vendor may have their own particular *view* on what access control means, i.e., to what is access being controlled. A vendor may provide access control to a company's internal network with a gateway device, or firewall. Another vendor may provide file level access control through *trusted* programs on the host computer.

In this chapter a sampling of industry products, three software products and one hardware device, has been analyzed. Key areas of analysis will revolve around reported functionality, architecture configurations, policy decision and policy enforcement points, and granularity of control. All products can be used with any of the leading web server and application server vendors, e.g., Microsoft, Solaris, and Linux, to name a few.

A. XMLACL

A product of XML Corporation, XMLAcl is a software-based server designed to provide web-based administration and access control over XML documents stored in a repository. XML Corporation claims on their web page that their product is the only software product that provides access control to individual products stored in native XML databases.

XMLAcl access control policies are based on *Owners*, *Users*, *Groups,* and *Others.* A system administrator configures roles and user groups for role-based access control. Individual users also have the power to permit and revoke privileges to other individual users or groups if they are the document *owner*. An owner cannot, however, define user groups. An access control list (ACL) for a document may contain any combination of an individual user, more than one user, and role-based groups of users.

Standard permission sets of *Read*, *Write*, and *Execute* are assigned or revoked by the owner or system administrator. As XMLAcl is web-based, access controls are set using a graphical user interface (GUI).

1. Architecture

XML documents are stored on a native XML database connected to the backend of the XMLAcl server. The server itself is the middle tier of a traditional three-tier architecture, and is co-located with a web server, as indicated by Figure 11. The network topology may introduce a firewall or demilitarized zone (DMZ) between the XMLAcl server and the Internet for additional protection.

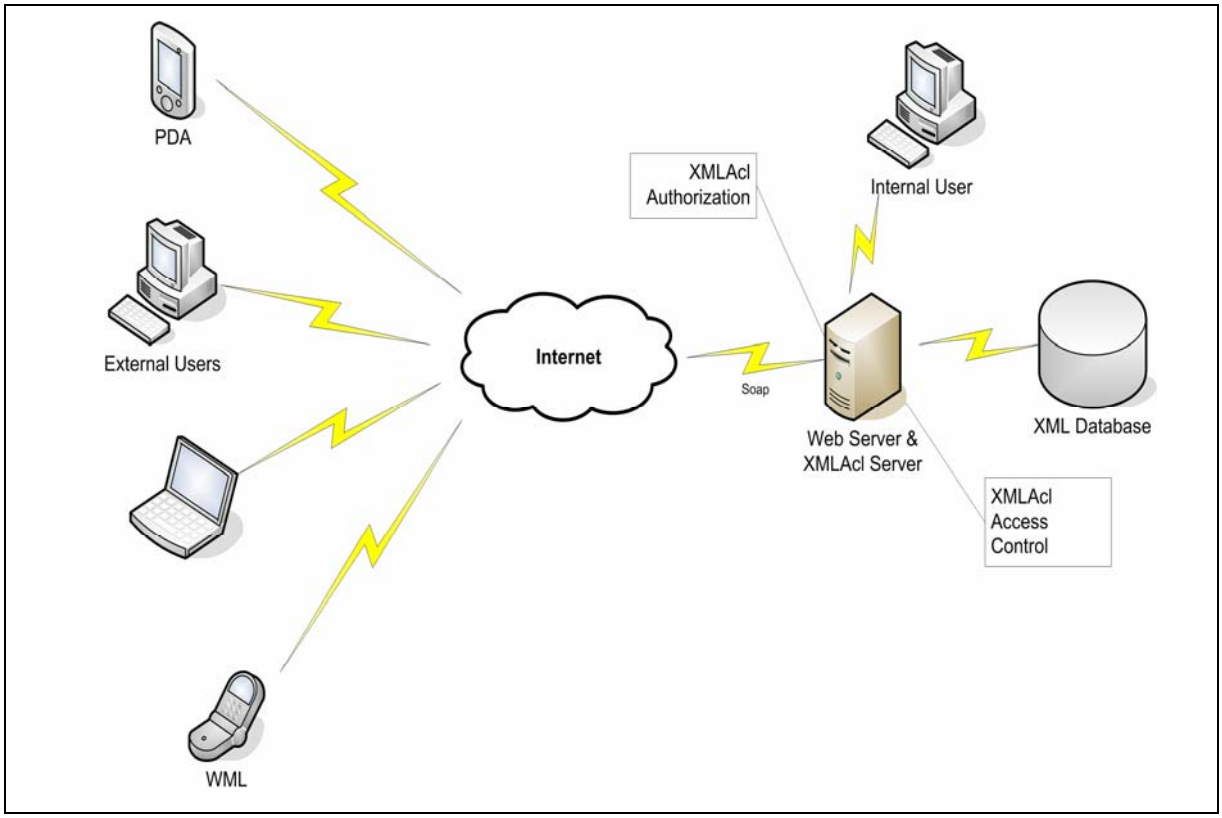

Figure 11. XMLAcl Architectural Configuration

2. Policy Decision and Policy Enforcement

The XMLAcl server is both the policy enforcement point and the policy decision point. The assumption, therefore, is that the XMLAcl server must be a trusted component. The server's first duty is to authorization users trying to gain access to the

data repository. A user, either inside the company's intranet, or from the Internet, connects to the XMLAcl server via hypertext transfer protocol (HTTP) or secure HTTP (HTTPS). At this point, the server authenticates the user, determines what groups and privileges the user has, and displays to the user the portions of the repository to which the user is allowed to access. The server then can begin processing requests for documents. As the XMLAcl server processes the requests, it determines whether or not access is granted, as well as, what actions are allowed. When a request is granted, the native XML database (Xindice, SleepyCat, eXist, etc.) is queried via XPATH. The proper document or document fragment is returned to the server where, if necessary, it can be changed into a requested format, such as PDF, CVS, text, or, PDA. The XML database does not serve as a decision or enforcement point. It simple acts as a repository.

3. Scale and Granularity

XMLAcl is designed to be used in a three-tier architecture. Configurations of this nature are capable of supporting hundreds of users concurrently.

One of the advantages of using XML is the ability to drill down into a document. XMLAcl works directly with a Native XML database. Therefore, queries can be as granular as elements and attributes. In addition, if a user has access to a document or group of documents, queries can be as focused as keyword searches. This is accomplished by converting the XML document to American Standard Code for Information Interchange (ASCII) text and then traversing through the entire document.

While XMLAcl is able to process XML queries at a highly granular level with respect to text, granularity for this survey is meant to measure the level of focus for access control purposes. By this definition, XMLAcl is not very granular. Access control is to the document or collection of documents. This creates an *all or nothing* situation. Upon gaining access to a document, a user has unlimited access to all data or information in those documents. If the user does not have access to a document, then the user does not have access to *any* data in the document.

B. DATAPOWER XS40 XML SECURITY GATEWAY

The Datapower XS40 XML Security Gateway is an out-of-the-box, drop-in network hardware device for a domain or enterprise. It runs on a proprietary *XG3* high speed XSLT/XML processor. The XS40's main purpose is to serve as a multifunction

XML gateway or router. Datapower's primary focus is web services security. The XS40 can be positioned on the edge of the network topology to perform as an XML firewall, a SOAP filtering service, and an access control device. There it examines and filters all XML message traffic entering or exiting the internal network. The firewall's responsibilities include checking messages to ensure they are well-formed. An XML message must have an end tag for every start tag and vice versa. If this is not the case, the message is mal-formed. A mal-formed message may indicate corrupted data, or result in a buffer overrun problem. In addition to being well-formed, some messages may be based on a schema. The firewall functions are able to validate that the schema is being met correctly. If these situations are not satisfied, the firewall throws the message out. SOAP filtering examines the SOAP headers and XML content to correctly route the message. Access control functions authorizes external users to gain access to internal systems, and can allow outbound transactions to be completed if they meet defined parameters. For example, if an outbound purchase order contains the following parameters, (1) it is for over $500, (2) digitally signed by the chief financial officer's (CFO) certificate, (3) targeted for vendor XYZ, and (4) is sent before 5 pm, it is allowed through. While and identical transaction sent after 5 pm will be rejected.[26] In this manner, the XS40 becomes an important policy enforcement point.

Additionally, the XS40 supports the XACML standard. If access control policy written with the XACML standard is used by the XS40, it can perform access control at a very granular level. This subject will be addressed in Section B.3.

1. Architecture

As shown in Figure 12 the XS40 is positioned at the edge of a company's network as the first line of defense for incoming XML web services transactions, and as the last line of control for outbound transactions.

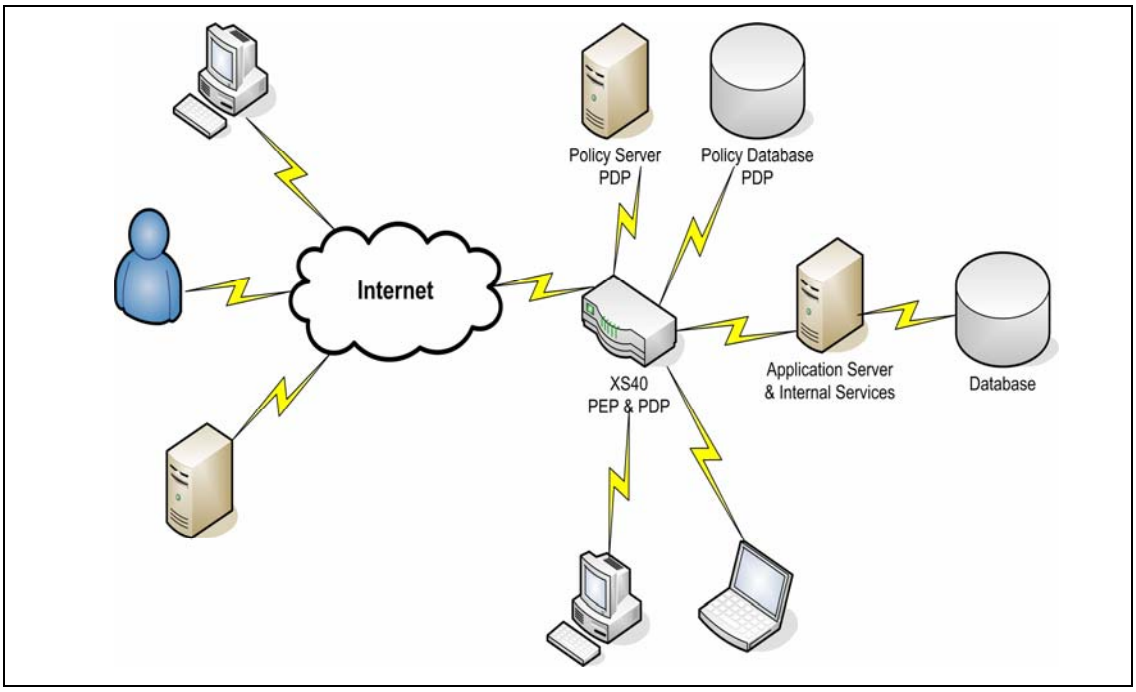

Figure 12. Datapower XS40 Conceptual Configureation

Configured in this topology, the XS40 can perform authorization decisions based on self-contained access control policies or, if integrated with policy servers and additional data stores, the XS40 is capable of enforcing fine-grained access controls.

2. Policy Enforcement Points and Policy Decision Points

The XS40 XML Security Gateway can serve as an all-in-one access control device. Its design allows for both the access control decisions to be made at the hardware, and to enforce those decisions. In other words, the XS40 can act as both the brains and the brawn. However, policy enforcement is the primary purpose of the XS40. In order for more detailed control of data and services, the XS40 must be integrated with a policy or access control server. This allows the XS40 to take advantage of XACML.

For example, suppose a doctor needs to review the medical records of a patient. The doctor submits a request via, HTTP, HTTPS, or SSL. The initial action taken for access into the network is authentication. The XS40 will authenticate the request based on a some number of parameters, ranging from passwords to URLs. The XS40 can then send an authorization request to the policy server (Policy Decision Point) for approval or rejection. The policy server's decision is sent back to the XS40 which will then either allow the transaction to take place, or reject the request.

Whether the XS40 is performing both PDP and PEP roles, or only the enforcement role, it is assumed to be a trusted component. If additional policy servers are being used to make policy decisions, they must also be trusted components.

3. Scale and Granularity

Datapower places strong emphasis on the fact that the XS40 is hardware and performs at *wirespeed.* This creates significant performance gains over software based access control (designed to perform similar tasks). Datapower's XML Generation 3 (XG3) processing technology is claimed to accelerate XML processing and prevent bottlenecks. An additional advantage gained is the ability to apply patches and updates to a single device, versus multiple application-based access control products.

The XS40 is not designed to provide granular access control over data and information. Its primary role is to allow or disallow web services transactions based on a combination of defined parameters such as passwords, URLs, and time of day. While that functionality is fairly robust, it is not designed to enforce fine-grained policy. If a user has met the correct parameters to access the company's file server, then that user may have unlimited access to the data on that server. However, when integrated with a policy server using XACML, the XS40 has the potential to enforce access down to more granular levels. XACML, for example, can define access control policy in a hierarchical manner, i.e., to specific nodes within a document. As earlier discussed, XML documents and data are formatted in a hierarchical manner. So if a patient's records are in an XML format, residing on a database, the policy may allow the doctor to access the patient's drug allergies, but not the patient's insurance information. Any attempt to access the patient's insurance information would be rejected by the XS40.

C. ENTRUST GETACCESS

GetAccess is described as a "high performance, scalable Web access control solution."[25] GetAccess is a software-based authentication and authorization server for web services and web portal access control and security. The GetAccess server authorizes based on RBAC and additionally authorizes transactions based on business rules, as well as, RBAC. Once a user has been properly identified (authentication), the roles of that user and which services that user has access to are determined.

While this scheme seems straightforward, there is a twist. The Entrust solution is divided between web portals, and web services. If a user is attempting to access portal servers, the GetAccess server is only responsible for authentication. After the user has been properly authenticated, the web server is responsible for all further access control for the session. If a user is attempting to access application servers, data repositories, or other web services devices, then the GetAccess server not only authenticates, it also performs authorization services i.e., access control.

1. Architecture

Entrust GetAccess architecture is divided into two tiers, a web server and the GetAccess server. The web server sits in-front of the GetAccess server, possibly in a DMZ, but always behind a firewall. A thin runtime agent is used to communicate between the servers (as shown in Figure 13). The runtime agent *intercepts* requests for access into the company's intranet and delivers them to the GetAccess server for a)authentication, and b) determination of what resources are being requested (e.g., portal or web services). The GetAccess server determines whether or not the request is from a current, authenticated session. If so, then two options are possible. First, if the resource requested is a portal resource, the runtime agent delivers the "permit" decision to the web server. The web server controls the session from that point on. Second, if the resources requested are web services, then the GetAccess Server performs access control on the request.

In this configuration, the GetAccess server must serve as the PDP for all actions, and assumes the additional role of PEP for web services transactions. The GetAccess server employs XACML-based policies to enhance or restrict access to portal resources, while utilizing SAML and proprietary policy services for web services transactions. Note that, Entrust documentation does not divulge the reason for separate policy services.

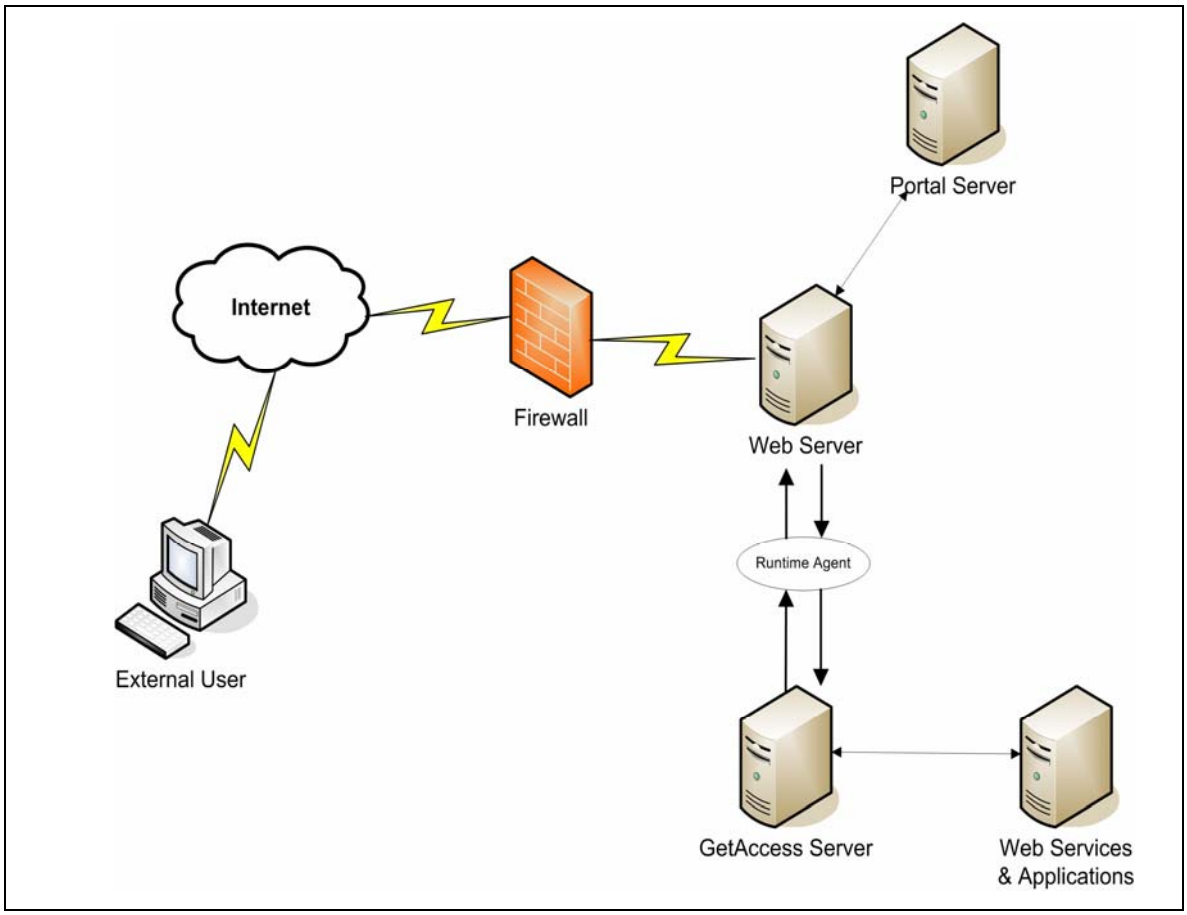

Figure 13. Entrust GetAccess Architecture

2. Policy Decision and Policy Enforcement

As described in Section C.1., the access control decisions are made at the GetAccess server. Based on the resources requested, i.e., portal or web services, policy enforcement is the responsibility of the web server for the former and the GetAccess server for the latter.

The GetAccess Server is made up of several components that handle the policy decision responsibilities.

a. Access Service

Access Service component handles authentication and authorization requests. Additionally, it can be configured to provide personalization based on user privileges and roles.

b. Entitlements Service

This component is the true *policy decision point* for GetAccess. This component determines which resources the users are allowed access to. The Entitlement Service compares roles of the user to the requested resources. It is in this component that the XACML based policies reside.

c. Logging Services

GetAccess servers provide the capability for detailed auditing of user sessions and system activity.

3. Scale and Granularity

The Entrust GetAccess product provides broad platform support across a large range of web servers and multiple language environments. Since clients do not require any special software, only a web browser, it scales well. Entrust claims their product delivers the performance and reliability to secure the largest web portals.

In addition to its ability to protect access to the internal network (course-grained access control), GetAccess has the capability to control fine-grained access to services and data. Based on XACML, access to portal resources is based on context sensitive policies. An example is the ability to restrict access to specific services based on the time of day or user roles. Additional policies within other applications can be integrated within the GetAccess server by using XACML. This allows the server to first allow access to the resource (i.e., file server), and then drill down to specific objects on that resource (i.e., a specific document).

D. SOFTWARE AG: TAMINO XML SERVER

Tamino XML Server is an XML-specific server for storing, managing, publishing and exchanging XML documents. Tamino keeps XML documents in the server's local data stores and in the document's *native* XML form. Software AG claims that this, in addition to supporting W3C XML standards, allow Tamino to control access to the element and attribute level of an XML document.

1. Architecture

A high level review of the Tamino architecture shows the server as the middle tier in a three tier configuration. As shown in Figure 14, the Tamino server processes HTTP and SOAP requests from the Internet and connects to the appropriate server or database.

The Tamino server is not involved in authentication or authorization for these *external* services. This results in the Tamino server basically directing traffic.

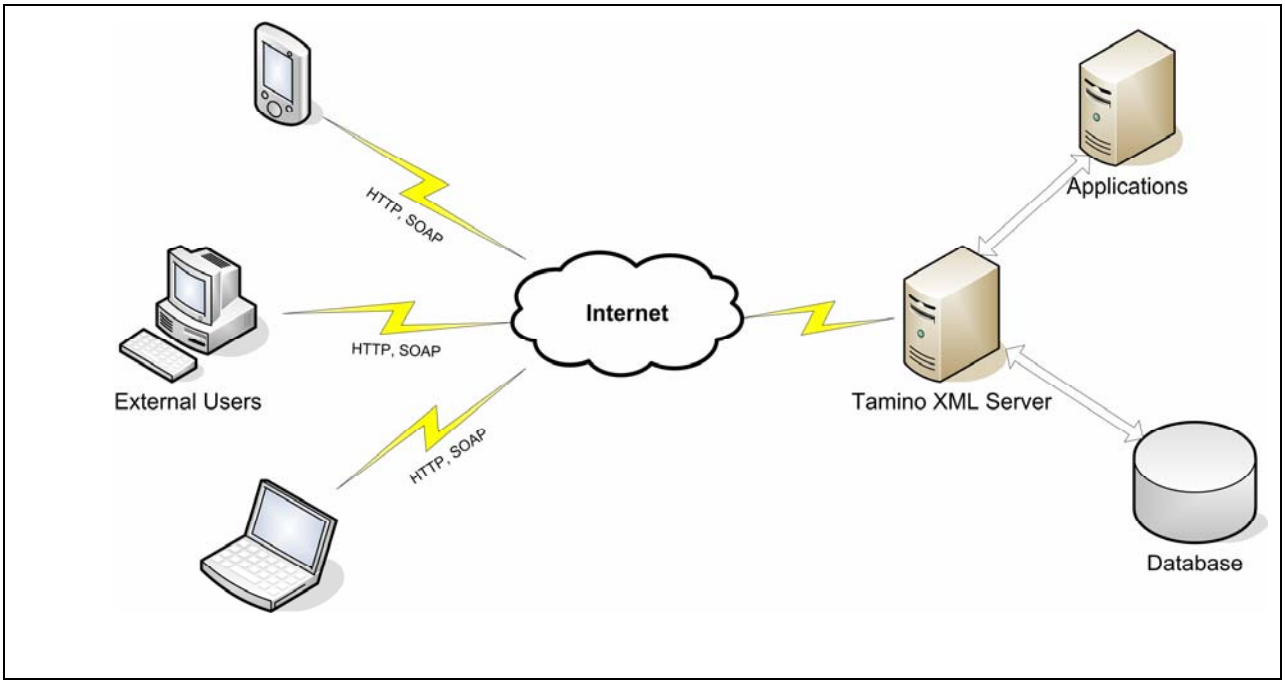

Figure 14. Software AG Tamino XML Server Architecture

However, inside the Tamino XML server, there is a lot of work going on. The server is composed of several components that provide core services and other enabling services. The core services are for specific XML server functionality, while the enabling services are for support and integration of application servers and external data sources.

As shown in Figure 15, there are five components in the core services area, two of which reside together. The *XML Engine* component is the central and most significant component for controlling and processing XML documents. The engine is responsible for efficient storing, querying, retrieval, and processing of XML documents. The *Data Map* component determines how XML objects, embedded in XML documents will be mapped to physical database structures and whether they will reside internally (i.e., data stores) or externally. *Tamino Manager* is the central point of Tamino XML Server's administration. This component is implemented as a client-server application that allows an administrator to manage the entire system over the web, to include database creation, server startup and shutdown, and backups. The *Security Manager*, which is co-located

with the Tamino Manager, is the component responsible for defining and modifying access rights to data stored in Tamino's data store(s). The Security Manager contains a GUI that is used to set up access control policies for document elements or attributes. The *Native XML Data Store* is part of the core components and separate from other databases. The Data Store is the physical hard disk in the Tamino server that stores the XML documents. Fine-grained access control can be applied to XML documents stored here. This will be further examined in section D-3.

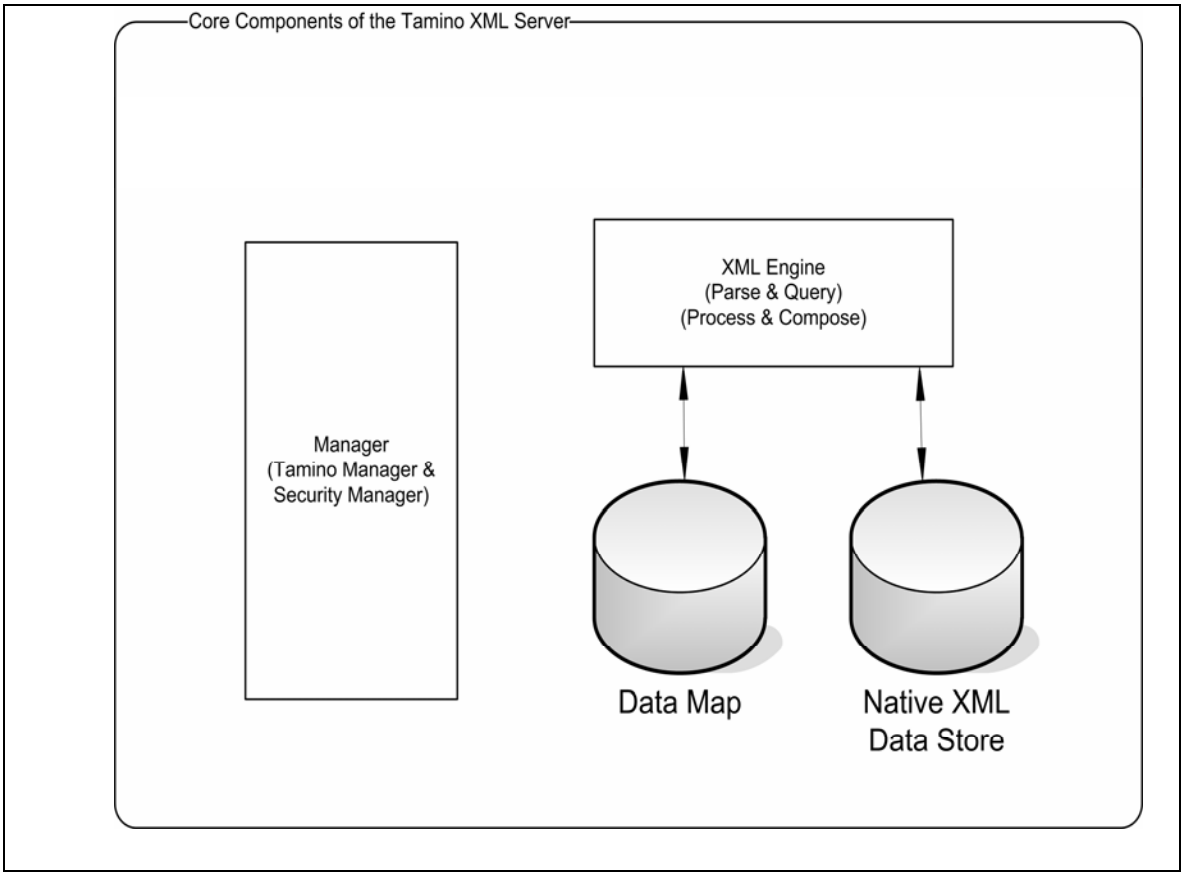

Figure 15. Tamino XML Server Core Components

The enabling services make Tamino XML Server easier to work with; however, since none of the components are directly involved in access control, they will not be covered.

2. Policy Decision and Policy Enforcement

The Tamino XML Server provides a server-side authorization check to grant or deny access to "secured" XML nodes. A secured XML node is an element or attribute in

an XML document stored *on* the Tamino XML Server. The Security Manager component of the core services defines and allows modifications of access rights for users or groups of users. There are four access rights, or *Authorization Levels* for secure nodes:

- No: Access to the node is denied
- Read: Read access is granted
- Change: Update access is granted
- Full: Define and undefined access is granted[30]

While, *No, Read, and Change* may be intuitive, *Full* means a user has the ability to change the access rights for that node. Typically, this privilege is reserved for security administrators. Each secure node (an XML element) has an attribute called the *Access Control Element (ACE)* added which specifies the authorization level. When processing requests for data, the XML Engine compares the user's authorization rights to the ACL stored on the server. Once the user has access to the document, any nodes secured with an ACE are reviewed to dictate the level of access on those secured nodes(i.e., *No, Read, Change, or Full)*. A user must have *Full* rights to add or change ACE's.

3. Scale and Granularity

Tamino XML Server was built to use W3C standards for XML. This allows it to be used across a wide variety of platforms and configurations. Additionally, Tamino has the ability to work with heterogeneous types of data, not just XML data. This allows the server to be configured with application servers and external databases. These factors support flexible solutions for businesses data and content management.

Tamino XML Server can be configured to provide access control to the document level for data in its external databases. Its capability for fine-grained access control is limited to its own data stores. In those stores, access control can be defined down to the XML node (element or attribute) which could be defined to achieve access control to individual words. However, with each ACE added to a secured node, there is overhead in the form of larger XML documents and larger ACL's. These issues will be address in Chapter IV.

IV. ANALYSIS OF XML ACCESS CONTROL ARCHITECTURES

A. DISCUSSION

All architectures reviewed in Chapter III provide some form of access control for XML formatted data. The commercial sector has influenced the development of access control in the form of authentication and authorization of web-based transactions involving XML documents and XML-based protocols (e.g., SOAP). Additionally, commercial products that use native XML databases as a means to store complete XML documents have developed access control schemes to protect data. Chapter III described how the application of access control, as well as topology and network configurations, vary between products. Chapter IV will provide an analysis of the products, pointing out high points and areas of concern.

1. Datapower XS40

The Datapower XS40 was the only hardware-based product reviewed in this thesis. Considering it is a hardware device, it actually stands out for its flexibility. The XS40 supports all major XML standards for security. Additionally, the XS40 can be integrated with products from most of the major IT vendors (e.g., Microsoft Active Directory, Sun Java System Manager, HP Open View, and more.) The XS40 can be integrated with external policy servers to establish it as a robust PEP. If those policies are written using XACML, granularity of access control can be quite fine, where fine granularity indicates access control to elements, attributes, or even distinct words in a document. However, it should be emphasized, without a well designed policy server, the XS40 is limited to coarse grained authentication services where the lowest level of access control is an entire document or collection of documents, XML firewall services, and router capabilities.

The XS40 has undergone the Common Criteria EAL-4 evaluation and has been certified by the Department of Defense for use as an XML security gateway.[24] Common Criteria evaluation gives the XS40 credibility as a high assurance product. None of the other products reviewed have an EAL designation.

2. XMLAcl

XMLAcl has a strength of simplicity. Its purpose is to manage XML documents and control who may access those documents. It has some flexibility in establishing either RBAC or user-based access controls. This is all done from the web browser of the user's choice. However, XMLAcl's most limiting factor is the lack of ability to establish ACL's at the element or attribute level. This creates an "all or nothing" situation regarding access to a document. Access controls to the element or attribute level would significantly improve this product. Additionally, other than SOAP, XMLAcl provides limited support for XML and web services standards. Most significantly absent is any support for XACML, which would allow for more granular and detailed access control policy. Support for XML-DSig would establish a method of verifying who created or changed a document, or verifying who transmitted a document. XML Sec is obviously required to guarantee confidentiality, especially if the XML documents carry trade secrets or proprietary information.

Lastly, XMLAcl serves as both the PDP and the PEP. This can be viewed as a single point of failure, or as an effort to keep the configuration simple.

3. Entrust GetAccess

Entrust GetAccess is mainly focused on authentication for web services transactions. The suggested configuration, which splits the responsibilities for PDP and PEP (web server for portals, GetAccess server for web services), may have its merits by separating *domains* based on importance. Additionally, its ability to support XML security standards allows it to be configured for fine-grained access control. The Entitlement Services component of the server is the key to robust access control, since this is were the access control policy resides. Surprisingly, the Entitlement Services does not utilize XACML, while the web server that controls access to the portals does use XACML.

Entrust GetAccess has been added to the list of approved E-Authentication products under the U.S. General Services Administration (GSA) E-Authentication Initiative.[25] Government agencies may employ GetAccess servers to authenticate on-line users and protect sensitive information.

4. Tamino XML Server

The Tamino XML Server's ability to control access to the element or attribute level of XML documents is its strength. However, with this advantage, comes a price, the requirement for a large amount of storage space, but because it stores XML documents in their native form, the Tamino server offers a fast rate of data retrieval.

Tamino's drawbacks include the limited ability for granular access control to external datbases and the lack of XML security standards it supports (most notably missing is XACML.) These missing standards limit Tamino's ability to guarantee integrity and confidentiality.

B. COMPARISON OF XML ACCESS CONTROL ARCHITECTURES

	DataPower XS40	**XML Corp XMLAcl**	**Entrust GetAccess**	**Tamino XML Server**
H/W Or S/W	H/W	S/W	S/W	S/W
Topology or Architectural Configuration	Multi-tiered	3-tier	2-tier	3-tier
Position in Topology or Architecture	Edge Device (firewall) Interface between Internet & Intranet	Middle Tier	Middle Tier	Middle Tier
Granularity Of Subjects	Group/Role Based	User Based Role Based	Role Based	User Based Role Based
Granularity Of Objects	Resources (inside the intranet)	Document	Resources (inside the intranet)	Document Element and Attribute level Ifor XML docs in data stores)
Location of Enforcement Points	XS40 Device	XMLAcl	Web Server \for Portal services. GetAccess Server for Web Services.	Tamino XML Server is the Enforcement Point
Location of Decision Points (Authentication)	XS40 Device	XMLAcl	GetAccess Server Runtime Agent	Tamino XML Server
Location of Decision Points (Access Control)	Backend Device Policy Server	XMLAcl	GetAccess Server Runtime Agent	Tamino XML Server

Table 1. Comparison of XML Access Control Architectures

A comparison of the architectures is provided in Table 1. The purpose of the table is to serve as a quick reference guide to the more important points of access control. Some areas of interest were omitted if there was no distinction or difference between the products, e.g., all products perform authentication on subjects. Most rows are intuitive, however, a brief narrative will be provided for clarity.

The first row lists whether the product is hardware or software. Other than the DataPower XS40, all the products are software based. This allows some flexibility of how to configure the system, e.g., dedicated access control server or co-locate with a web server.

The next area of interest deals with where the access control hardware or software is physically located in a network topology or logically located in a tiered architecture. While the DataPower XS40 is intended to be physically located at the edge of a network, once again, the software products show the flexibility of a dedicated server or collocation with the web server. When viewing the software products from a tiered standpoint, it is interesting to see how the Entrust GetAccess server moves away from the more conventional three-tier architecture the other software products suggest. Once the GetAccess server determines the request is for portal services, it can release control of those transactions to the web server, thereby freeing it up for requests for more sensitive resources.

The level of access control is directly related to how granular a system defines the subjects trying to access objects, as well as, how granular a system defines the objects. The table indicates that all products are capable of allowing an administrator to set up groups or roles that can then be assigned access privileges. Two of the products, XMLAcl and Tamino XML Server, have the capability to assign access control down to a single user.

The other half of this equation, is how detailed are access controls to objects. When dealing with standard relational databases or files on a file server, the question of granularity is quite straightforward. Either you have access to a tuple of the database or you don't. Either you have access to a file or you don't. Additional detail follows if you do have access to that object, e.g., read, write, and execute. Due to XML's hierarchical

structure, there is added complexity to granularity of objects. An object may be a document or an element defined by XML tags in that document. XMLAcl currently only provides access controls to a document. Each document is considered an object. Once a user gains access to that object, the user has access to all the contents of the object.

The DataPower XS40 and the Entrust GetAccess server can be configured with XACML to generate access controls as granular as the elements and attributes in the document itself are defined. Using XACML, an administrator can create extremely detailed ACL's, in the form of an XML document and store it on a policy server. The PDP then uses these XACML policies to make decisions. While XACML provides significant granularity, the cost for such detail is the time to define the policy and storing the policy documents.

The final area of interest, while comparing these products, is to analyze where in the architecture enforcement and decisions take place. Enforcement is quite straightforward. Either a transaction or request is allowed or denied. However, since these products involve web services, decisions must be made on authentication of the user (i.e., should the user gain access to the system or to send data out of the system) and decisions must be make on access to information and documents (i.e., should an authenticated user be allowed access to a particular document or piece of data.)

While Table 1 does not possess the detail of Chapter III, it does provide a quick reference to each product side-by-side. Areas of most significance, due to differences, are Granularity of Subjects, Granularity of Objects, and the locations of Decision Points and Enforcements Points. For example, if reviewing the Granularity of Subjects, it can be quickly observed that only two of the products provide granularity to the individual user. Additional observations can be made from row to row.

THIS PAGE LEFT INTENTIONALLY BLANK

V. CONCLUSION

The objective of this thesis was to survey and analyze what security architectures are currently being employed to protect XML-based data in the commercial industry. As the US government, military, and intelligence agencies continue to rely on COTS products, it is imperative that there is an understanding of what those products are capable of doing, as well as, what are the liabilities of employing them.

This thesis established a starting point for that understanding. However, there is much work that needs to follow. Three areas of follow-on work are implementation and testing of current products, implementation and testing of current products with XACML used to create fine grained policies, and finally there should be an in-depth analysis of XACML in general. This would include efficiency testing, vulnerability testing, and examining how XACML can be leveraged to meet strict MAC-based requirements.

XACML seems to be the key to a standard-driven solution that is fine-grained. Solutions that do not use XACML as a policy decision point tool lack the ability to control access at individual elements. Without that fine-grained access control, products are not much more effective than current access control lists. The selling point that the products have in either case is the ability to drill down into a document with the help of XML, which has nothing to do with security.

All of the products analyzed have a foundation of access control based on web services authorization to enter the service domain. If an external user is authorized to enter their domain (or an internal user is authorized to access external services from within the domain), then the product has met the main goal for access control. The use of XACML-based policy decision points strengthen the ability of the XS40 and GetAccess products to establish true access control to objects. Therefore, these products should be given further analysis.

If XML is to be used to enable safe and secure sharing of sensitive information, more work must be accomplished to guarantee the ability to control data.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. G. Bush. “Executive Order 13356 of 27 August 2004”, Retrieved 20 September 2005 from http://www.fas.org/irp/offdocs/eo/eo-13356.htm. Last accessed 20 September 2005.

2. Intelligence Community Metadata Working Group, Intelligence Information Sharing Standards. Retrieved 20 September 2005 from https://www.icmwg.org/iiss/introduction.asp. Last accessed 20 September 2005.

3. R. Bourret. “XML and Databases”. Retrieved May 2005 from http://www.rpbourret.com/xml/XMLAndDatabases.htm.

4. J. Shanmugasundaram, K. Tuft, G. He, C. Zhang, D. DeWitt, and J. Naughton. “Relational Databases for Querying XML Documents: Limitations and Opportunities”. In VLDB, Edinburgh, Scotland, September 1999

5. E. X. DeJesus, “XML Enters the DBMS Arena”. *Computerworld*, October 2000.

6. D. Florescu and D. Kossmann. “Storing and Querying XML Data Using an RDBMS”. *IEEE Data Eng. Bulletin,* 22(3):27-34, September 1999.

7. B. Steegmans, R. Bourret, O. Cline, O. Guyennet, S. Kulkarni, S. Priestly, V. Sylenko, U. Wahli. “XML for DB2 Information Integration”. IBM International Technology Support Organization, July 2004,

8. J. Clark and S. DeRose (Eds). “XML Path Language (XPath) Version 1.0”. W3C Recommendation, November 1999. Retrieved May 2005 from http://www.w3.org/TR/xpath.

9. S. Boag, D. Chamberlin, M. F. Fernandez, J. Robie, and J. Simeon (Eds). “XQuery 1.0: An XML Query Language” W3C Working Draft, April 2005. Retrieved May 2005 from http://www.w3.org/TR/2005/WD-xquery-20050404/.

10. R. Sandhu and F. Chen. “The Multilevel Relational (MLR) Data Model”. *IEEE Trans. On Information and System Security (TISSEC),* 1(1), 1998.

11. K. Staken. “Introduction to Native XML Databases”. , October 2001. Retrieved May 2005 from www.xml.com/pub/a/2001/10/31/nativexmldb.html

12. M. Kay. “XML Databases”. Software AG, Retrieved May 2005 from http://www.xmlstarterkit.com/xmlzone/WP_XML_Databases_E.pdfMarch 2003

13. B. Steegmans, R. Bourret, O. Cline, O. Guyennet, S. Kulkarni, S. Priestly, V. Sylenko, U. Wahli. “XML for DB2 Informatio Integration”. Retrieved September 2005. http://www.redbooks.ibm.com/redbooks/pdfs/sg246994.pdf.

14. D. Mertz. "XML Matters: Putting XML in Context with Hierarchical, Relational, and Object-oriented Models". Webpage April 2001. http://www-106.ibm.com/developerworks/library/x-matters8/index.html.

15. Deep Set Operators for XQuery, Bo Luo, Dongwon Lee, Wang-Chien Lee, Peng Liu, In *ACM SIGMOD Workshop on XQuery Implementation, Experience and Perspectives (XIME-P)*, Baltimore, MD, USA, June 2005

16. QFilter: Fine-Grained Run-Time XML Access Control via NFA-based Query Rewriting, Bo Luo, Dongwon Lee, Wang-Chien Lee, Peng Liu, In *13th ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, Washington DC, USA, November 2004

17. A Flexible Framework for Architecting XML Access Control Enforcement Mechanisms, Bo Luo, Dongwon Lee, Wang-Chien Lee, Peng Liu, In *VLDB Workshop on Secure Data Management in a Connected World (SDM)*, Toronto, Canada, August 2004

18. Supporting XML Security Models using Relational Databases: A Vision, Dongwon Lee, Wang-Chien Lee, Peng Liu, In *XML Database Symposium (XSym)*, Berlin, Germany, September 2003

19. E. Simon, P. Madsen, & C. Adams. "An Introduction to XML Digital Signatures." Retrieved June 2005 from http://www.xml.com/lpt/a/2001/08/08/xmldsig.html.

20. Department of Defense, "Trusted Computer Security Evaluation Criteria," *DoD 5200.28-STD*, 1985.

21. http://csrc.nist.gov/rbac/NIST-TL-RBAC-bulletin.html. "An Introduction to Role-Based Access Control." NIST/ITL Bulletin, December 1995. A web site. Last viewed on 25 July 2005

22. L. Anathamurthy. "Introduction to Web Services."Retrieved July 2005 from http://www.developer.com/services/article.php/1485821.

23. D. Hunter, K. Cagle, C. Dix, R. Kovack, J. Pinnock, J, Rafter. Beginning XML 2nd Edition. Wiley Publishing, Inc. Indianapolis, IN. 2003

24. J. Merrells. "XACML: XML Access Control". XML Europe Conference 2004. Retrieved August 2005 from http://www.idealliance.org/papers/dx_xmle04/papers/04-01-04/04-01-04.html

25. M. Mactaggart. "Enabling XML Security." 01 September 2001. "Retrieved July 2005 from http:www-128.ibm.com/developerworks/security/library/s-xmlsec.html)

26. Datapower Newsroom. Retrieved on 18 August 2005 from http://www.datapower.com/newsroom/pr_070605_gsa.html

27. Entrust website, retrieved August 2005 from http://www.entrust.com/internet-access-control/index.htm.

28. P. Madsen & E. Maler, Editors. “SAML V2.0, Executive Overview, Committee Draft 01”, 12 April 2005. Electronic copy retrieved from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security. Last accessed 16 Sep 2005.

29. Datapower XS40 Access Control Product Description. Retrieved from http://www.datapower.com/products/accesscontrol.html. Last accessed on 16 Sep 2005.

30. Tamino Technical Details/Security Management. Retrieved from http://www1.softwareag.com/Corporate/products/tamino/prod_info/tech_details/td_sec_mgmt.asp. Last accessed on 16 Sep 2005.

THIS PAGE LEFT INTENTIONALLY BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. AFIT/CI
   Wright-Patterson AFB, OH

4. Hugo A. Badillo
   NSA
   Fort Meade, MD

5. George Bieber
   OSD
   Washington, DC

6. RADM Joseph Burns
   Fort George Meade, MD

7. John Campbell
   National Security Agency
   Fort Meade, MD

8. Deborah Cooper
   DC Associates, LLC
   Roslyn, VA

9. CDR Daniel L. Currie
   PMW 161
   San Diego, CA

10. Louise Davidson
    National Geospatial Agency
    Bethesda, MD

11. Vincent J. DiMaria
    National Security Agency
    Fort Meade, MD

12. LCDR James Downey
NAVSEA
Washington, DC

13. Dr. Diana Gant
National Science Foundation
Washington, DC

14. Jennifer Guild
SPAWAR
Charleston, SC

15. Richard Hale
DISA
Falls Church, VA

16. LCDR Scott D. Heller
SPAWAR
San Diego, CA

17. Wiley Jones
OSD
Washington, DC

18. Russell Jones
N641
Arlington, VA

19. David Ladd
Microsoft Corporation
Redmond, WA

20. Dr. Carl Landwehr
National Science Foundation
Arlington, VA

21. Steve LaFountain
NSA
Fort Meade, MD

22. Dr. Greg Larson
IDA
Alexandria, VA

23. Penny Lehtola
NSA

Fort Meade, MD

24. Ernest Lucier
Federal Aviation Administration
Washington, DC

25. CAPT Deborah McGhee
Headquarters U.S. Navy
Arlington, VA

26. Dr. Vic Maconachy
NSA
Fort Meade, MD

27. Doug Maughan
Department of Homeland Security
Washington, DC

28. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

29. John Mildner
SPAWAR
Charleston, SC

30. Jim Roberts
Central Intelligence Agency
Reston, VA

31. Charles Sherupski
Sherassoc
Round Hill, VA

32. Dr. Ralph Wachter
ONR
Arlington, VA

33. David Wirth
N641
Arlington, VA

34. Daniel Wolf
NSA
Fort Meade, MD

35. Jim Yerovi
NRO
Chantilly, VA

36. CAPT Robert Zellmann
CNO Staff N614
Arlington, VA

37. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

38. Timothy E. Levin
Naval Postgraduate School
Monterey, CA

39. Maj Mark Estlund
Naval Postgraduate School Student
Papillion, NE